Magnet Project API Documentation

Route Authentication & Role Protection Summary

Module	Route/Action	Public	Auth Required	Role(s) Required
Products	GET /api/products	Yes	No	None
	POST /addProductsByBusiness	No	Yes	business
	POST /addProductsByMagnet_employee	No	Yes	magnet_employee, admin
	PUT /products/:id	No	Yes	business (own), admin, magnet_employee
	DELETE /products/:id	No	Yes	business (own), admin, magnet_employee
	PUT /products/:id/approve	No	Yes	admin, magnet_employee
	PUT /products/:id/decline	No	Yes	admin, magnet_employee
Categories	GET /api/categories	Yes	No	None
	POST/PUT/DELETE /categories	No	Yes	business (own), admin, magnet_employee
Orders	POST /api/orders	No	Yes	customer
	GET /api/orders/my	No	Yes	customer
	GET /api/orders/:id	No	Yes	admin, magnet_employee, customer (own), business (if owns product in order)
	GET /api/orders	No	Yes	admin, magnet_employee
	PUT /api/orders/:id/status	No	Yes	admin, magnet_employee
Reviews	POST /products/:id/reviews	No	Yes	customer
	GET /products/:id/reviews	Yes	No	None
	DELETE /reviews/:id	No	Yes	admin, magnet_employee
Wishlist	All /api/wishlist	No	Yes	customer
Addresses	All /api/addresses	No	Yes	customer
User	/api/user/profile	No	Yes	customer
	/api/user/business-requests	No	Yes	admin, magnet_employee

Module	Route/Action	Public	Auth Required	Role(s) Required
	/api/user/business-approval	No	Yes	admin, magnet_employee
	/api/user/business/:id	No	Yes	admin, magnet_employee
	/api/user/business-profile	No	Yes	business, admin, magnet_employee
Auth	Registration/Login/OTP	Yes	No	None
	Forgot Password	No	Yes	Any authenticated user

Notes

- **Auth Required**: Requires a valid JWT token in the `Authorization` header.
- **Role(s) Required**: Requires the user to have one of the listed roles. If "(own)" is specified, the user must own the resource (e.g., their own product or order).
- **Public**: Anyone can access, no authentication required.
- **All**: All HTTP methods (GET, POST, PUT, DELETE) for that endpoint.
- **Business (if owns product in order)**: Business user can access if any product in the order is owned by them (see order tracking section).

Each module's section below should reference this table for route protection. For details on request/response, see the rest of the documentation.

Authentication & User Management APIs

Message Format

All API responses now return bilingual messages in both English and Arabic:

```
{
  "status": "success",
  "message": {
    "en": "User registered successfully",
    "ar": "تم تسجيل المستخدم بنجاح"
  },
  "data": { ... }
}
```

Verification Rules

Phone Number Verification

- **If phone number is provided**: Only the phone is verified (`isPhoneVerified = true`, `isEmailVerified = false`)
- **If no phone number is provided**: Only the email is verified (`isEmailVerified = true`, `isPhoneVerified = false`)

This applies to both customer and business registration.

Auth Routes

1. Register User

- **POST** `/api/auth/register`
- **Description:** Register a new customer user with automatic verification based on phone number presence.
- **Body:**
  - `firstname` (string, required)
  - `lastname` (string, required)
  - `email` (string, required)
  - `phone` (string, optional)
  - `password` (string, required)
  - `country` (string, required)
  - `language` (string, optional, default: 'en')
- **Verification Logic:**
  - If `phone` is provided: `isPhoneVerified = true`, `isEmailVerified = false`
  - If `phone` is not provided: `isEmailVerified = true`, `isPhoneVerified = false`
- **Response:**
  - `201 Created`: User info + JWT token with verification status

---

2. Register Business

- **POST** `/api/auth/business-register`
- **Description:** Register a new business user (requires approval) with automatic verification based on phone number presence.
- **Body:**
  - `firstname`, `lastname`, `email`, `password`, `crNumber`, `vatNumber`, `companyName`, `companyType`, `country`, `city`, `district`, `streetName`, `phone` (all required)
- **Verification Logic:**
  - If `phone` is provided: `isPhoneVerified = true`, `isEmailVerified = false`
  - If `phone` is not provided: `isEmailVerified = true`, `isPhoneVerified = false`
- **Response:**
  - `201 Created`: Business info (under review) with verification status

---

3. Send Email OTP

- **POST** `/api/auth/send-email-otp`
- **Description:** Send an OTP to a new email (fails if email already exists).
- **Body:**
  - `email` (string, required)
- **Response:**
  - `200 OK`: Success message (bilingual)

---

4. Send Phone OTP

- **POST** `/api/auth/send-phone-otp`
- **Description:** Send an OTP to a new phone (fails if phone already exists).
- **Body:**
  - `phone` (string, required)
- **Response:**
  - `200 OK`: Success message (bilingual)

---

5. Confirm OTP

- **POST** `/api/auth/confirm-otp`
- **Description:** Confirm an OTP for any email or phone identifier (works for both registered and non-registered users).
- **Body:**
  - `identifier` (string, required)
  - `otp` (string, required)
- **Response:**
  - `200 OK`: OTP verified successfully (bilingual message)

---

6. Confirm Login OTP

- **POST** `/api/auth/confirm-login-otp`
- **Description:** Confirm an OTP for registered users and return user data with JWT token (like login API).
- **Body:**
  - `identifier` (string, required)
  - `otp` (string, required)
- **Response:**
  - `200 OK`: User info + JWT token (bilingual message)
  - `404 Not Found`: User not found (bilingual message)

---

7. Login

- **POST** `/api/auth/login`
- **Description:** Login with email/phone and password.
- **Body:**
  - `identifier` (email or phone, required)
  - `password` (string, required)

- **Response:**
  - `200 OK`: User info + JWT token (bilingual message)

---

8. Login with OTP

- **POST** `/api/auth/login-with-otp`
- **Description:** Request an OTP for login (email or phone).
- **Body:**
  - `identifier` (email or phone, required)
- **Response:**
  - `200 OK`: OTP sent (bilingual message)

---

9. Forgot Password

- **POST** `/api/auth/forgot-password`
- **Description:** Change password for authenticated user.
- **Headers:** `Authorization: Bearer <token>`
- **Body:**
  - `newPassword` (string, required)
- **Response:**
  - `200 OK`: Password updated (bilingual message)

Product Routes

1. Get Products

- **GET** `/api/products`
- **Description:** Get all approved products (public, any user can call). Admin/magnet_employee can see all products.
- **Response:**
  - `200 OK`: List of products
- **Product Object:**
  - `code` (string, auto-generated if not provided, e.g. "A001")
  - `category` (string, required)
  - `name` (string, required)
  - `images` (array of strings)
  - `description` (string)
  - `color` (string)
  - `features` (string)
  - `unit` (string)
  - `minOrder` (number)
  - `pricePerUnit` (string)
  - `stock` (number)
  - `accessories` (string)
  - `customFields` (array of objects, min 5, max 10, each: `{ key, value }`)
  - `status` (string: 'pending', 'approved', 'declined')

  - `owner` (user id, required)
  - `approvedBy` (user id, admin/employee who approved)
  - `createdAt` (date)

---

2. Add Product (Business)

- **POST** `/api/products/addProductsByBusiness`
- **Description:** Business user adds a new product (pending approval). Product code is auto-generated if not provided.
- **Headers:** `Authorization: Bearer <token>`
- **Body:**
  - `category` (string, required)
  - `name` (string, required)
  - `images` (array of strings, optional)
  - `description`, `color`, `features`, `unit`, `minOrder`, `pricePerUnit`, `stock`, `accessories` (all optional)
  - `customFields` (array of objects, min 5, max 10, each: `{ key, value }`, required)
- **Response:**
  - `201 Created`: Product info (pending approval) with bilingual message

---

3. Add Product (Magnet Employee)

- **POST** `/api/products/addProductsByMagnet_employee`
- **Description:** Magnet employee adds a new product (approved immediately). Product code is auto-generated if not provided.
- **Headers:** `Authorization: Bearer <token>`
- **Body:**
  - `category` (string, required)
  - `name` (string, required)
  - `images` (array of strings, optional)
  - `description`, `color`, `features`, `unit`, `minOrder`, `pricePerUnit`, `stock`, `accessories` (all optional)
  - `customFields` (array of objects, min 5, max 10, each: `{ key, value }`, required)
  - `owner` (user id, required)
- **Response:**
  - `201 Created`: Product info (approved) with bilingual message

---

4. Update Product

- **PUT** `/api/products/:id`
- **Description:** Business user updates their own product (pending approval). Admin/magnet_employee can update and approve/decline.
- **Headers:** `Authorization: Bearer <token>`
- **Body:**
  - Any product field (see above)

  - `status` (optional, only admin/magnet_employee can set to 'approved' or 'declined')
- **Response:**
  - `200 OK`: Updated product info with bilingual message

---

5. Delete Product

- **DELETE** `/api/products/:id`
- **Description:** Business user deletes their own product, or admin/employee deletes any product.
- **Headers:** `Authorization: Bearer <token>`
- **Response:**
  - `200 OK`: Product deleted with bilingual message

---

6. Approve Product

- **PUT** `/api/products/:id/approve`
- **Description:** Admin/magnet_employee approves a product.
- **Headers:** `Authorization: Bearer <token>`
- **Response:**
  - `200 OK`: Product approved with bilingual message

---

7. Decline Product

- **PUT** `/api/products/:id/decline`
- **Description:** Admin/magnet_employee declines a product.
- **Headers:** `Authorization: Bearer <token>`
- **Response:**
  - `200 OK`: Product declined with bilingual message

Real-Time Order Tracking (WebSocket)

Overview

- The API supports real-time order status updates using WebSocket (Socket.IO).
- Clients can subscribe to updates for specific orders and receive instant notifications when the order status changes (e.g., confirmed, shipped, delivered).

How It Works

1. **Client authenticates with a JWT token when connecting to the WebSocket.**
2. **Client joins a room for a specific order using the order ID.**
3. **When the order status changes, the server emits an `orderStatusUpdate` event to all clients in that room.**

WebSocket Connection (Socket.IO)

- **URL:** `ws://<your-server>:5000` (or `wss://` for HTTPS)

- **Library:** socket.io-client

Client Example (JavaScript/React):

```
import { io } from 'socket.io-client';
const socket = io('http://localhost:5000', {
  auth: { token: '<JWT_TOKEN>' }
});

// Join the order room
socket.emit('joinOrderRoom', '<ORDER_ID>');

// Listen for updates
socket.on('orderStatusUpdate', (data) => {
  // data: { orderId, status, statusLog, updatedAt }
  console.log('Order update:', data);
});
```

Order Room Access Rules

- **Admin & magnet_employee:** Can track any order.
- **Business:** Can track orders that include products they own.
- **Customer:** Can track only their own orders.
- Unauthorized attempts to join order rooms will be ignored.

Server-Side Security

- The server authenticates each socket connection using the JWT token.
- The server enforces the above access rules for joining order rooms.

Order Status Update Event

- **Event:** `orderStatusUpdate`
- **Payload:**

```
{
  "orderId": "...",
  "status": "shipped",
  "statusLog": [
    { "status": "pending", "timestamp": "..." },
    { "status": "confirmed", "timestamp": "..." },
    { "status": "shipped", "timestamp": "..." }
  ],
  "updatedAt": "2024-07-10T12:00:00.000Z"
}
```

Security Notes

- Clients must provide a valid JWT token when connecting.
- The server will only allow joining order rooms for orders the user is authorized to track (see access rules above).
- Unauthorized attempts to join rooms will be ignored.

(Other modules remain as previously documented, unless you want to update them as well)

Response Examples

Success Response

```
{
  "status": "success",
  "message": {
    "en": "Product added successfully",
    "ar": "تمت إضافة المنتج بنجاح"
  },
  "data": {
    "product": {
      "code": "A001",
      "category": "Electronics",
      "name": "Smartphone",
      "images": ["https://example.com/image1.jpg"],
      "description": "A great phone",
      "color": "Black",
      "features": "Waterproof, Dual SIM",
      "unit": "piece",
      "minOrder": 1,
      "pricePerUnit": "1000",
      "stock": 50,
      "accessories": "Charger, Earphones",
      "customFields": [
        { "key": "Warranty", "value": "2 years" },
        { "key": "Origin", "value": "Japan" },
        { "key": "Battery", "value": "4000mAh" },
        { "key": "Screen", "value": "6.5 inch" },
        { "key": "Weight", "value": "180g" }
      ],
      "status": "pending",
      "owner": "60f7b3b3b3b3b3b3b3b3b3b3",
      "approvedBy": null,
      "createdAt": "2024-07-10T12:00:00.000Z"
    }
  }
}
```

Error Response

```
{
  "status": "error",
  "message": {
    "en": "Must provide 5-10 custom fields",
    "ar": "يجب توفير 5 إلى 10 حقول مخصصة"
  }
}
```

Notes

- All endpoints require authentication unless stated otherwise.
- Role-based access is enforced for some endpoints (Admin, Employee, Business, Customer).
- For endpoints that update or approve/decline, validation middleware is used.
- **All API responses now include bilingual messages in English and Arabic.**
- **Verification status is automatically set during registration based on phone number presence.**
- **Product codes are auto-generated in the format A001, A002, ... if not provided.**
- **Products require 5–10 custom fields (key/value pairs).**
- **The GET and PUT `/api/user/profile` endpoints are protected by the `requireCustomer` middleware, allowing only users with the 'customer' role to access them.**

For more details or example requests, refer to the code or request further examples.